STRUTS 2 SENDING EMAIL

This chapter will teach you how yYou can send an email using your Struts 2 application. For this exercise, you need to download and install the mail.jar from JavaMail API 1.4.4 and place the **mail.jar** file in your WEB-INF\lib folder and then proceed to follow the standard steps of creating action, view and configuration files.

Create Action:

The next step is to create an Action method that takes care of sending the email. Let us create a new class called **Emailer.java** with the following contents.

```
package com.tutorialspoint.struts2;

import java.util.Properties;
import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import com.opensymphony.xwork2.ActionSupport;

public class Emailer extends ActionSupport {

   private String from;
   private String password;
   private String to;
   private String subject;
   private String body;

   static Properties properties = new Properties();
   static
   {
      properties.put("mail.smtp.host", "smtp.gmail.com");
      properties.put("mail.smtp.socketFactory.port", "465");
      properties.put("mail.smtp.socketFactory.class",
                     "javax.net.ssl.SSLSocketFactory");
      properties.put("mail.smtp.auth", "true");
      properties.put("mail.smtp.port", "465");
   }

   public String execute()
   {
      String ret = SUCCESS;
      try
      {
         Session session = Session.getDefaultInstance(properties,
            new javax.mail.Authenticator() {
            protected PasswordAuthentication
            getPasswordAuthentication() {
            return new
            PasswordAuthentication(from, password);
            }});

         Message message = new MimeMessage(session);
         message.setFrom(new InternetAddress(from));
         message.setRecipients(Message.RecipientType.TO,
            InternetAddress.parse(to));
         message.setSubject(subject);
         message.setText(body);
         Transport.send(message);
      }
      catch(Exception e)
```

```
        {
            ret = ERROR;
            e.printStackTrace();
        }
        return ret;
    }

    public String getFrom() {
        return from;
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getTo() {
        return to;
    }

    public void setTo(String to) {
        this.to = to;
    }

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }

    public static Properties getProperties() {
        return properties;
    }

    public static void setProperties(Properties properties) {
        Emailer.properties = properties;
    }
}
```

As seen in the source code above, the **Emailer.java** has properties that correspond to the form attributes in the email.jsp page given below. These attributes are

- **from** - The email address of the sender. As we are using Google's SMTP, we need a valid gtalk id
- **password** - The password of the above account
- **to** - Who to send the email to?
- **Subject** - subject of the email
- **body** - The actual email message

We have not considered any validations on the above fields, validations will be added in the next chapter. Let us now look at the execute() method. The execute() method uses the javax Mail library to send an email using the supplied parameters. If the mail is sent successfuly, action returns SUCCESS, otherwise it returns ERROR.

Create main page:

Let us write main page JSP file **index.jsp**, which will be used to collect email related information mentioned above:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Email Form</title>
</head>
<body>
   <em>The form below uses Google's SMTP server.
   So you need to enter a gmail username and password
   </em>
   <form action="emailer" method="post">
   <label for="from">From</label><br/>
   <input type="text" name="from"/><br/>
   <label for="password">Password</label><br/>
   <input type="password" name="password"/><br/>
   <label for="to">To</label><br/>
   <input type="text" name="to"/><br/>
   <label for="subject">Subject</label><br/>
   <input type="text" name="subject"/><br/>
   <label for="body">Body</label><br/>
   <input type="text" name="body"/><br/>
   <input type="submit" value="Send Email"/>
   </form>
</body>
</html>
```

Create Views:

We will use JSP file **success.jsp** which will be invoked in case action returns SUCCESS, but we will have another view file in case of an ERROR is returned from the action.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Email Success</title>
</head>
<body>
   Your email to <s:property value="to"/> was sent successfully.
</body>
</html>
```

Following will be the view file **error.jsp** in case of an ERROR is returned from the action.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Email Error</title>
```

```
</head>
<body>
   There is a problem sending your email to <s:property value="to"/>.
</body>
</html>
```

Configuration Files:

Now let us put everything together using the struts.xml configuration file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

   <constant name="struts.devMode" value="true" />
   <package name="helloworld" extends="struts-default">

      <action name="emailer"

         method="execute">
         <result name="success">/success.jsp</result>
         <result name="error">/error.jsp</result>
      </action>

   </package>

</struts>
```

Following is the content of **web.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
   >

   <display-name>Struts 2</display-name>
   <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>

   <filter>
      <filter-name>struts2</filter-name>
      <filter-class>
         org.apache.struts2.dispatcher.FilterDispatcher
      </filter-class>
   </filter>

   <filter-mapping>
      <filter-name>struts2</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
</web-app>
```

Now, right click on the project name and click **Export > WAR File** to create a War file. Then deploy this WAR in the Tomcat's webapps directory. Finally, start Tomcat server and try to access URL http://localhost:8080/HelloWorldStruts2/index.jsp. This will give you following screen:

Enter the required information and click **Send Email** button. If everything goes fine then you should see the following

page